课后作业二：概率分类法

任务：使用贝叶斯估计或 MLE（最大似然估计），来预测鸢尾花数据集中花的种类。

数据集：鸢尾花数据集是统计学和机器学习中用于分类的经典数据集。该数据集包含了三种不同的鸢尾花：Setosa、Versicolor 和 Virginica，每种各 50 个样本。每个样本有四个属性：萼片长度、萼片宽度、花瓣长度和花瓣宽度，所有的测量单位都是厘米。数据集根据 4:1 的比例划分为训练集和测试集。概率分类法是一种基于概率理论的方法，适合处理此类分类问题。

实现：

1.导入必要的库

```
from sklearn.model_selection import train_test_split     #机器学习中非常重要的库，包括一些分类、
                                                         回归、聚类、降维、模型选择和预处理
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as plt   #可视化工具
import pandas as pd     #用来分析结构化数据
import numpy as np      #提供高性能的矩阵运算
import csv    #读写文件的库
```

2.导入训练数据并提取特征值和目标值

"sepal length (cm)", "sepal width (cm)", "petal length (cm)", "petal width (cm)"为特征值，"species"为目标值

```
iris_data = pd.read_csv(r'D:\dataenclorse\second\iris_train.csv')
X = iris_data[["sepal length (cm)", "sepal width (cm)", "petal length (cm)", "petal width
(cm)"]].values
y = iris_data["species"].values
```

3.划分数据

```
# 构造训练数据和测试数据
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
```

4.构造 knn 模型并训练该模型

```
# 构造 KNN 模型
knn = KNeighborsClassifier(n_neighbors=1)


# 训练模型
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
```

5.加载 iris_test.csv 的数据并对 iris_test.csv 进行预测

```
# 做出预测
file_path = ("D:\dataenclorse\second\iris_test.csv")
```

```
data = pd.read_csv(r'D:\dataenclorse\second\iris_test.csv', usecols=["sepal length (cm)","sepal
width (cm)","petal length (cm)","petal width (cm)"])
grouped_data = [data.iloc[i:i+1].values.tolist() for i in range(0, len(data), 1)]
grouped_data = [sum(sublist, []) for sublist in grouped_data]

X_new = np.array(grouped_data)
prediction = knn.predict(X_new)
print("预测的目标类别是：{}".format(prediction))
file_path=r'D:\dataenclorse\second\test.csv'
```

6.预测结果与加载的数据一起保存到 test.csv 文件中
getdata 和 getdata2 函数与作业一中一模一样，仅仅有读取文件的内容不同

```
def getdata(path):
    data_frame = pd.read_csv(r'D:\dataenclorse\second\iris_test.csv')  # skiprows=14
    data_x,data_y = np.array(data_frame['sepal length (cm)']), np.array(data_frame['sepal width
(cm)'])
    return data_x,data_y
def getdata2(path):
    data_frame = pd.read_csv(r'D:\dataenclorse\second\iris_test.csv')  # skiprows=14
    data_p,data_q= np.array([data_frame['petal length (cm)'], np.array(data_frame['petal width
(cm)'])])
    return  data_p,data_q

data_x,data_y=getdata('iris_test.csv')
data_p,data_q=getdata('iris_test.csv')
with open(file_path,'w',encoding='utf-8',newline='') as f:
        fieldnames=['sepal length(cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width
(cm)','class']
        f_csv = csv.DictWriter(f, fieldnames=fieldnames)
        f_csv.writeheader()
        for i in range(0,len(prediction)):
            f_csv.writerow({'sepal length(cm)':data_x[i],'sepal width (cm)':data_y[i],'petal
length (cm)':data_p[i],'petal width (cm)':data_q[i],'class':prediction[i]})
```

7. 结果如下：

```
(pytorch) D:\dataenclorse>python -u "d:\dataenclorse\second\second.py"
预测的目标类别是：[1 0 2 2 1 0 1 2 1 1 1 0 0 0 0 1 2 1 2 2 0 2 0 1 2 2 2 2 0 0]
```

A	B	C	D	E	F
sepal len	sepal wid	petal len	petal wid	class	
5.8162431	2.550183	5.8162431	2.550183	1	
5.8100076	4.3979506	5.8100076	4.3979506	0	
8.1375468	3.0778532	8.1375468	3.0778532	2	
5.2453112	2.943514	5.2453112	2.943514	2	
6.9100196	3.0875745	6.9100196	3.0875745	1	
5.1617493	3.9913823	5.1617493	3.9913823	0	
5.5034111	3.1224913	5.5034111	3.1224913	1	
6.8831622	3.0830352	6.8831622	3.0830352	2	
6.4285034	1.9282514	6.4285034	1.9282514	1	
6.0816364	2.4118773	6.0816364	2.4118773	1	
6.0735042	2.9357557	6.0735042	2.9357557	1	
4.1280984	3.4539907	4.1280984	3.4539907	0	
5.9342591	3.9596878	5.9342591	3.9596878	0	
5.1763616	3.204539	5.1763616	3.204539	0	
5.3239954	3.8175243	5.3239954	3.8175243	0	
6.4081433	3.8030474	6.4081433	3.8030474	1	
7.3972944	2.6968336	7.3972944	2.6968336	2	
5.006388	1.6894347	5.006388	1.6894347	1	
6.1919004	2.1313759	6.1919004	2.1313759	2	
6.1093193	2.3811147	6.1093193	2.3811147	2	
4.3330253	3.303213	4.3330253	3.303213	0	
6.2076257	3.1410137	6.2076257	3.1410137	2	
4.9362235	3.6321525	4.9362235	3.6321525	0	
6.6849524	2.8475112	6.6849524	2.8475112	1	
7.5248511	3.7826922	7.5248511	3.7826922	2	
6.5936434	2.7914832	6.5936434	2.7914832	2	
6.6942635	2.1277791	6.6942635	2.1277791	2	
7.1895987	2.2001647	7.1895987	2.2001647	2	
4.7193036	2.5926936	4.7193036	2.5926936	0	
4.5589682	3.2530239	4.5589682	3.2530239	0	